Literature Review – XCS224U – Fine-tuning vs prompting in adaptation of language model for structured text generation

Sergey Grebenkin
AI Professional Program
Stanford Online Course
XCS224U
sergey.grebenkin@gmail.com

Georgii Shushuev
AI Professional Program
Stanford Online Course
XCS224U
shyshyev@gmail.com

Abstract

The growing use of natural language understanding (NLU) for process automation has led to increased interest in methods that allow machines to interpret and execute complex instructions given in free text. One specific use case in the medical field is the automatic interpretation of doctors' prescriptions, which requires converting unstructured text into a structured format that machines can act upon. While fine-tuning pre-trained language models has been the traditional approach to adapting models for specific tasks like classification or named entity recognition (NER), prompting has emerged as a flexible alternative, particularly with the advent of large language models (LLMs) like GPT-4.

This literature review compares the efficacy of fine-tuning and prompting for structured output tasks that do not involve free-form text generation, such as converting textual prescriptions into predefined grammar or structured data formats like JSON. We explore whether fine-tuning consistently outperforms prompting or if a well-engineered prompt can yield competitive or even superior results without the need for extensive labeled datasets or model retraining. Our analysis focuses on key NLP tasks like sentiment analysis, multi-label classification, and information extraction, questioning whether prompting alone can rival fine-tuning in non-generative contexts.

The review also addresses the trade-offs between these methods and their implications for future NLP applications. The findings aim to guide researchers and practitioners in selecting the most appropriate approach for tasks where structured, non-generative outputs are required.

1 General problem / Task definition

Natural language understating very often is an essential task for process automation. When we are talking about "automation" we are talking about something that can be performed without humans. At our job, we faced a task when it was necessary to understand what the doctor asked to perform in the prescription. There could be many different wishes about the ways how the doctor wants the treatment to be performed. To automatically create such treatment, or to automatically check if the treatment was created properly we need to make the machine understand what the doctor asks. So we need to create some structured language to explain the machine doctor's wishes.

Although there are many different preferences for how the doctor wants the treatment to be carried out, their number is finite, and the parameters of each preference are also finite. So if we can create some grammar that can describe any doctor's wishes, it means that we can make the machine understand what the doctor asks. The task here is only to create an algorithm that can convert free text doctor's prescriptions to this grammar.

In the case of humans, it takes several years of education to be able to do it. In the case of machines, it should be comprehensive instruction that explains all the ways how doctors can ask for some actions. Or it should be labeled a dataset of examples of how free text prescriptions convert to the grammar? Or both? What model is more effective to create such a conversion algorithm? It is the questions that we faced and this literature review has a goal to find the answer. It is important to remark that it is a quite broad problem that is actual almost in any field where free text instructions exists. Even if we imagine the AI hairdresser the client, if he hasn't a photo what result does he want to get, will explain in a free text way how he wants his hair cut.

In the domain of natural language processing (NLP), there are multiple strategies for leveraging large language models (LLMs) to solve a variety of tasks. Two key approaches are fine-tuning and prompting. While fine-tuning has traditionally been the go-to method for adapting pre-trained models to specific tasks, prompting has gained considerable attention due to its simplicity and flexibility, especially when working with very large language models like GPT-4.

This literature review focuses on tasks that do not require free-form text generation but instead involve more structured outputs, such as classification, named entity recognition (NER), information extraction, and text-to-structured-data transformations (e.g., converting unstructured text into a JSON format - the very grammar that was mentioned earlier). These tasks often require models to identify patterns, label data, or map textual inputs to predefined structures, which can be approached in several ways.

Historically, tasks like sentiment analysis or multilabel-multiclass classification have been effectively solved by fine-tuning models such as BERT on task-specific datasets. Fine-tuning allows models to adapt deeply to the task at hand by adjusting all their parameters based on the labeled data available. However, this approach can be challenging when datasets are scarce or tasks are highly domain-specific.

In such cases, prompting presents an alternative. With a well-designed prompt, even a general-purpose LLM (e.g., GPT-4) can be leveraged to produce high-quality annotations for tasks where large labeled datasets do not exist. For example, one might craft a detailed prompt that instructs the model on how to label text or convert it into structured data (the grammar). This prompt-based approach can serve as a data augmentation technique, where the LLM is used to generate a synthetic dataset, which can then be used to fine-tune a smaller, task-specific model.

The key question this review seeks to explore is whether fine-tuning can consistently outperform prompting in tasks where no free-text generation is required. If a well-engineered prompt can achieve high accuracy without the need for extensive dataset creation and model retraining, then prompt-based methods could offer a more efficient path to solving classification and extraction problems. However, it remains unclear whether improving prompts alone can lead to better performance than fine-tuning, especially as prompt engineering requires considerable trial and error, and the potential of fine-tuning remains powerful when sufficient task-specific data is available.

Thus, this review will examine the comparative advantages of each approach in non-generative tasks, particularly focusing on questions like:

- Can fine-tuning always outperform prompting when data is available, or can prompt-based LLM methods be equally effective?
- Are there specific tasks or conditions where prompting could replace the need for fine-tuning entirely?
- What trade-offs exist between these methods, and how might this influence the choice of approach in future NLP applications?

2 Summary of Articles

2.1 Language Models are Few-Shot Learners

We would like to start with an article that, by its appearance, indicated that large language models can compete with fine-tuned state-of-the-art solutions. At the time of writing the paper, one of the authors (Jared Kaplan) worked at OpenAI and is now Chief Science Officer at Anthropic (Language Models are Few-Shot Learners, 2020).

2.1.1 Key Points

Research Objective: The study demonstrates that scaling up language models significantly improves their performance in few-shot learning tasks, sometimes even becoming competitive with fine-tuning approaches.

GPT-3 Model: The authors trained GPT-3, an autoregressive language model with 175 billion parameters, which is 10 times larger than previous models. GPT-3 shows strong results on many NLP datasets, including translation, question-answering, and close tasks.

Training Methods:

- Zero-Shot: The model performs tasks without any examples, using only a textual description of the task.
- One-Shot: The model is given one example of the task before performing it.

- Few-Shot: The model is given several examples of the task before performing it.
- Fine-Tuning: updates the weights of a pre-trained model by training on thousands of supervised labels specific to the desired task.

Results:

- GPT-3 shows significant performance improvements with increased model size.
- In few-shot tasks, GPT-3 sometimes outperforms models fine-tuned on specific tasks.
- On some tasks, GPT-3 still lags behind fine-tuned models, especially in zero-shot and one-shot settings

2.1.2 Mentions of BERT

Most interesting for our goal is a comparison with the fine-tuned BERT model:

- The paper compares GPT-3's performance with BERT-Large on the SuperGLUE dataset. BERT-Large was fine-tuned on the SuperGLUE training set (125K examples), as well as on MultiNLI (392K examples) and SWAG (113K examples), totaling 630K fine-tuning examples.
- GPT-3's performance in the few-shot setting is compared to BERT-Large and BERT++, showing that GPT-3 with one example in context (one-shot) and eight examples (few-shot) achieves results comparable to fine-tuned BERT models.

Results on SuperGLUE:

- On SuperGLUE tasks, GPT-3 in the few-shot setting shows results close to state-of-the-art, held by fine-tuned models like BERT-Large.
- In tasks such as COPA and ReCoRD, GPT-3 achieves near-SOTA performance in one-shot and few-shot settings, falling only slightly short of the fine-tuned T5 model with 11 billion parameters.

Limitations and Weaknesses:

- The paper notes that GPT-3 sometimes falls short of fine-tuned models like BERT on tasks requiring the comparison of two sentences or text snippets (e.g., WiC and RTE).
- GPT-3 also shows weaker results on tasks that benefit from bidirectional architectures, which is a strength of BERT.

2.1.3 Conclusion

At the time of writing, approaches based on BERT had already been developed long ago and were very strong, and the GPT-3, as we now know, was far from being the smartest model, and even it had already achieved the quality of the state-of-the-art models in some tasks after just a few shots.

2.2 Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm

The next article is about the prompting strategy, namely that a zero-shot is better than a few-shot. This article is interesting for us because we reached the same conclusion independently in our task while working with the prompt (Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm, 2021).

2.2.1 Key Points

Research Objective: The study explores methods for effectively controlling and evaluating large language models like GPT-3 through prompt programming, emphasizing the potential of zero-shot prompts over few-shot prompts.

Prompt Programming: The authors argue that few-shot examples often serve to locate an already learned task rather than teaching the model the task during runtime. They propose that prompt programming, especially using zero-shot prompts, can be more effective.

In this article added the one new training method:

- Metaprompt Programming: Introducing the idea of a metaprompt that seeds the model to generate its own natural language prompts for a range of tasks.

Results:

- Zero-Shot Prompts: The study finds that zero-shot prompts can match or even exceed the performance of few-shot prompts with minor prompt engineering.
- Task Location: Few-shot examples primarily help the model locate the task within its existing knowledge rather than learning it anew.
- Prompt Engineering: Simple changes in prompt formatting can significantly improve performance, indicating that GPT-3's zero-shot capabilities were underestimated.

2.2.2 Conclusion

The study emphasizes the importance of prompt programming in leveraging the full potential of large language models. It suggests that zero-shot prompts and metaprompt programming can significantly enhance the performance and flexibility of models like GPT-3, potentially surpassing the need for few-shot learning in many cases. The authors call for further research into automated prompt generation and more sophisticated benchmarking methods to better evaluate and utilize these models.

2.3 Fine-tuning and prompt engineering for large language models-based code review automation

This paper (Fine-tuning and prompt engineering for large language models-based code review automation, 2024) is interesting for us because consists detailed review and comparison of using LLM techniques in a quite close task for us.

2.3.1 Key Points

Research Objective: The study investigates the performance of code review automation using large language models (LLMs) based on two approaches: fine-tuning and prompt engineering.

Methods:

- Fine-Tuning: Further training the model on a specific code review dataset.
- Prompt Engineering: Providing explicit instructions to guide the model's generation process without requiring a specific code review dataset.

2.3.2 Results

Fine-Tuning:

- Fine-tuning GPT-3.5 with zero-shot learning achieves 73.17%–74.23% higher Exact Match (EM) compared to the approach by Guo et al.
- Fine-tuning GPT-3.5 with zero-shot learning achieves 63.91%–1100% higher EM compared to non-fine-tuned GPT-3.5.

Prompt Engineering:

- GPT-3.5 with few-shot learning achieves 46.38%–659.09% higher EM compared to zero-shot learning.
- Using a persona in prompts reduces EM by 1.02%–54.17%.

2.3.3 Conclusions

- Fine-Tuning: Recommended for achieving the highest performance in code review automation.
- Few-Shot Learning: When data is insufficient for fine-tuning (e.g., cold-start problem), few-shot learning without a persona should be used.

2.4 Large Language Models for Text Classification: From Zero-Shot Learning to Instruction-Tuning

One of the most obvious tasks for comparison prompted LLM with fine-tuned BERT-like models is text classification. But LLMs can also be compared with each other based on how they cope with this task and which approaches to their use are best suited for this (Large Language Models for Text Classification: From Zero-Shot Learning to Instruction-Tuning, 2024). The article is fresh, includes a comparison with GPT-4o and even here BERT-like approaches for classification show themselves to be on the level.

2.4.1 Introduction

The article explores the application of large language models (LLMs) to supervised text classification, focusing on stance detection. It compares ten models ranging from 86 million to 1.7 trillion parameters across four training regimes: zero-shot learning, few-shot learning, fine-tuning, and instruction-tuning.

2.4.2 Key Findings

Zero-shot Learning:

- Advanced LLMs can perform text classification without any task-specific training by leveraging their pre-trained knowledge.
- GPT-4o achieved high accuracy in zero-shot settings, outperforming many fine-tuned models.

Few-shot Learning:

- Adding a few labeled examples alongside prompts can sometimes improve performance but is highly sensitive to the choice of exemplars.

- Performance varies significantly depending on the exemplars provided.

Fine-tuning:

- Fine-tuning smaller models can be competitive due to their relatively high accuracy and low cost.
- Larger models like GPT-3 Davinci showed significant improvements with fine-tuning, especially with more training data.

Instruction-tuning:

- Instruction-tuned models can handle complex tasks by combining the strengths of prompting and fine-tuning.
- Llama3-70B showed comparable performance to GPT-4o after instruction-tuning.

Results:

- Zero-shot and Few-shot Learning: Larger models like GPT-4o and Llama3-70B performed exceptionally well in zero-shot settings.
- Fine-tuning: Smaller models like BERT and DeBERTa showed competitive performance when fine-tuned on larger datasets.
- Instruction-tuning: Llama3-70B achieved high accuracy in complex tasks involving structured data like comment-reply threads.

Recommendations

- Model Selection: Depends on the task complexity, document length, number of documents, and available resources.
- Zero-shot/Few-shot Learning: Suitable for small datasets or when labeled data is scarce.
- Fine-tuning: Effective for larger datasets with sufficient annotated examples.
- Instruction-tuning: Ideal for complex tasks requiring detailed instructions and handling structured data.

2.4.3 Conclusion

LLMs offer powerful tools for text classification, with zero-shot and instruction-tuning opening new possibilities for social scientific research. The choice of model and training regime should be guided by the specific requirements of the task and available resources.

2.5 Fine-tuning after Prompting: an Explainable Way for Classification

In the paper Zhezhong Wang at al. investigates a hybrid approach to use the pre-trained language models (PLMs) for classification tasks, combining the strengths of both fine-tuning and prompting. The authors introduce a method called F&P, that ”attains performance on par with Fine-tuning while preserving the outstanding explainability inherent to Prompting methods”. In this method, a prompt is used to guide the PLM’s predictions, and a linear classifier is fine-tuned on the model’s output to improve classification accuracy. The classifier’s weights are then used to create a ”verbalizer,” mapping the model’s output words to their corresponding classes. Authors also introduce an optimisation of F&P method called Fine-tuning and AUTOPROMPT (F&AP) leveraging AUTOPROMPT (Shin et al., 2020).

The key advantage of F&P is that both the prompt and verbalizer are based on natural language, making the process more explainable to humans compared to traditional fine-tuning methods. Experiments on English and Chinese datasets, using benchmarks such as GLUE and CLUE, demonstrate that the F&P and F&AP methods achieve superior performance with fine-tuning in terms of avg F1 score while maintaining greater interpretability. Additional parameters are not added to the model, therefore the size is not changed. Furthermore, the authors argue that this approach offers a balance between explainability and accuracy, suggesting its potential in practical applications where model transparency is essential.

2.6 No More Fine-Tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence

In this article Wang at al. investigate the use of prompt tuning as an alternative to traditional fine-tuning methods for code intelligence tasks. The study measures the performance of prompt tuning and fine-tuning on three tasks: defect detection, code summarization, and code translation, using pre-trained models such as CodeBERT and CodeT5. Prompt tuning modifies model inputs by adding task-specific prompts, allowing it to align more closely with the model’s pre-training objectives.

The experiments show that prompt tuning consistently outperforms fine-tuning across all tasks, particularly in low-resource scenarios where data is scarce. It also significantly improves BLEU scores in code summarization by over 26%. Different prompt templates and verbalizer choices can affect the performance, and prompt tuning shows more advantages with smaller pre-trained models.

The authors suggest that prompt tuning is a promising method for code intelligence, especially in cases with limited training data, and recommend further research into optimizing prompt design for various tasks.

2.7 Comparison of Prompt Engineering and Fine-Tuning Strategies in Large Language Models in the Classification of Clinical Notes

In the article Zhang at al. evaluate different methods to classify clinical notes, particularly focusing on identifying metastatic cancer. They compare the effectiveness of prompt engineering and fine-tuning strategies across several language models, including GPT-3.5, GPT-4, and Llama-7B, against BERT-based models and human annotators.

The study shows that GPT-4 outperformed GPT-3.5 and Llama-7B in classification tasks, demonstrating superior accuracy and robustness across different prompts and token sizes. Prompt engineering, especially structured prompts, showed strong performance in zero-shot learning scenarios, often outperforming fine-tuning models.

Fine-tuning Llama-7B yielded moderate results, but did not surpass GPT-4's performance, particularly in tasks requiring precision. One-shot learning provided little to no advantage over zero-shot learning in most cases.

GPT-4's performance remained stable even when keywords were removed or token counts reduced, showing resilience in handling incomplete or sparse data.

The article concludes that using effective prompt engineering, large language models like GPT-4 can potentially replace domain-specific models like PubMedBERT in biomedical tasks, offering comparable or superior performance without extensive fine-tuning.

3 Compare and Contrast

In the above-mentioned articles the fine-tuning and prompting are considered two important methods for models adaptation for the given task. Mainly pre-trained models were used in the studies. The results are dependent on the task and the models used. However, there are some general conclusions that could be done based on them.

While majority of the articles highlight a prompting as the superior methods for enhancing the model performance, some of the studies show that fine-tuning could be a better option in some specific applications. Pornprasit et al. proves fine tuning achieves better exact match score compared to non fine-tuned models. Few-shot prompting shows improvement in specific context.

It should be noted that there is some contradiction in the articles by Reynolds et al. (2021) and Pornprasit et al. (2024), one says that the few-shot prompting is better than the zero-shot prompting, while in the other the key idea is that if done correctly the zero-shot is no worse or even better than the few-shot. Probably this could be attributed to the different years of studies and some conclusions from Reynolds et al. (2021) could not be applied to modern models.

One of the approaches that is worth mentioning is the combined usage of both fine-tuning and prompting called F&AP, evaluated by Wang et al. While not being a game changer, for some applications it could be the best option as it leads to avg F1-score improvement on GLUE from 1% for RoBERTa-based models up to 3.6% for OpenAI GPT.

As it turns out, fine-tuning could be a good option when using pre-trained model for a specific task. Also, On the other side, fine-tuning could be computationally expensive in some situations and requires labeled datasets. Every time it's recommended to fit the model for a new type of task separately.

To sum up, the pros of fine-tuning are deep adaptation to specific task, high-performance on well-defined datasets for specific domain. The cons are necessity of labeled datasets and computational complexity. The pros of prompting are quick implementation, large datasets are not required. On the other site the cons could be high dependence of prompt quality.

4 Future work

We are going to prepare a solution for for conversion of text to json representation in medical domain. Therefore based on analysis of the articles, the following options would be a good direction for further research:

1. Using hybrid approach for text to sequence conversion. This is to leverage the strengths of both fine-tuning and prompting approaches.

2. Domain specific prompt engineering. This could be the best option to maximize the performance of LLMs such as Chat GPT or Gemini in specialized fields.

3. Extend the above-mentioned studies and include more modern LLMs from other vendors such Gemini and with slightly different architectures and others.

4. Evaluate the fine-tuning approach for less powerful LLMs. This could be a good approach for on-premise solutions when one considers hosting the model on the local resources.

5. Real-time adaptation to the task. When having a dynamic environment when tasks and requirements frequently change.

These directions could deepen our understanding of fine-tuning versus prompting and enhance possible applications. By following the mentioned directions we can continue to improve the performance of the models.

References

Brown, Tom and Mann, Benjamin and Ryder, Nick and Subbiah, Melanie and Kaplan, Jared D and Dhariwal, Prafulla and Neelakantan, Arvind and Shyam, Pranav and Sastry, Girish and Askell, Amanda and Agarwal, Sandhini and Herbert-Voss, Ariel and Krueger, Gretchen and Henighan, Tom and Child, Rewon and Ramesh, Aditya and Ziegler, Daniel and Wu, Jeffrey and Winter, Clemens and Hesse, Chris and Chen, Mark and Sigler, Eric and Litwin, Mateusz and Gray, Scott and Chess, Benjamin and Clark, Jack and Berner, Christopher and McCandlish, Sam and Radford, Alec and Sutskever, Ilya and Amodei, Dario 2020. Language Models are Few-Shot Learners *arXiv:2005.14165*

Laria Reynolds, Kyle McDonell 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm *arXiv:2102.07350*

Chanathip Pornprasit, Chakkrit Tantithamthavorn 2024. Fine-tuning and prompt engineering for large language models-based code review automation *arXiv:2402.00905*

Youngjin (YJ) Chae, Thomas Davidson 2024. Large Language Models for Text Classification: From Zero-Shot Learning to Instruction-Tuning *SocArXiv. doi:10.31235/osf.io/sthwk.*

Zezhong Wang , Luyao Ye , Hongru Wang, Boyang Xue, Yiming Du, Bin Liang, Kam-Fai Wong 2024. Fine-tuning after Prompting: an Explainable Way for Classification. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 133–142, Bangkok, Thailand. Association for Computational Linguistics.

Zhang X, Talukdar N, Vemulapalli S, Ahn S, Wang J, Meng H, Murtaza SMB, Leshchiner D, Dave AA, Joseph DF, Witteveen-Lane M, Chesla D, Zhou J, Chen B. 2024. Comparison of Prompt Engineering and Fine-Tuning Strategies in Large Language Models in the Classification of Clinical Notes *medRxiv Preprint. 2024 Feb 8:2024.02.07.24302444. doi: 10.1101/2024.02.07.24302444.*

Chaozheng Wang, Yuanhang Yang, Cuiyun Gao, Yun Peng, Hongyu Zhang, Michael R. Lyu 2024. No More Fine-Tuning? An Experimental Evaluation of Prompt Tuning in Code Intelligence *arXiv:2207.11680*

Taylor Shin, Yasaman Razeghi, Robert L Logan IV 2020. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235.*